Abstraction

Inheritance (Is-a) Relationship

Association (Has-a / Part-of) Relationships

Polymorphism (Overloading / Overriding)

Data Hiding (Private)

Encapsulation (Setter + Getter + Private)

A class is said to be tightly encapsulated if every data member declared as private. Whether the class contains getter & setter methods are not and whether those methods declared as public or not these are not required to check

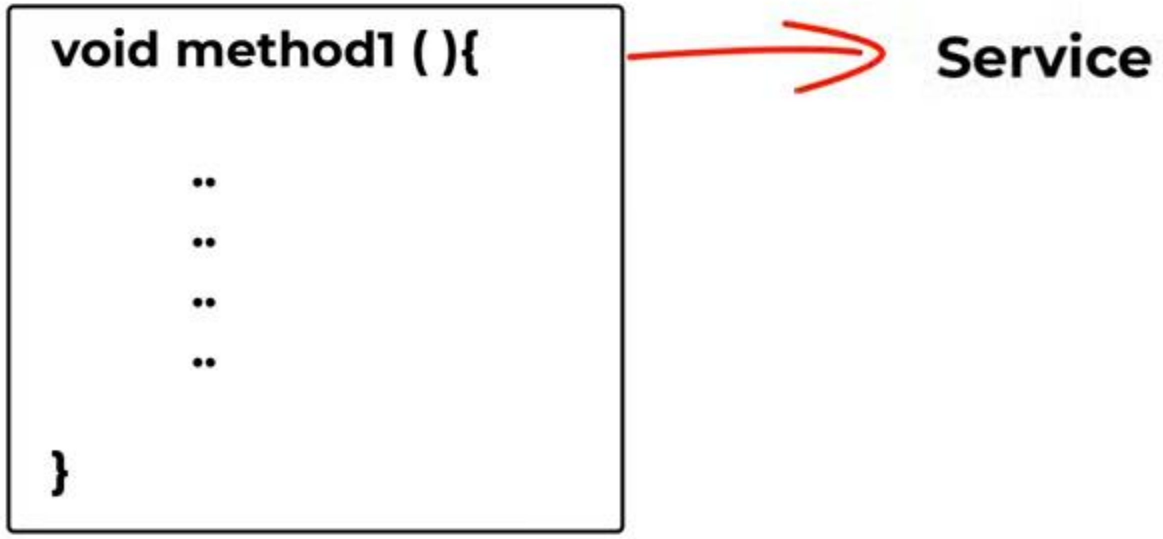
Code Reusability ← **Abstraction** → **Security**

Abstraction

What is the Abstraction

إخفاء التنفيذ الداخلي　　　　　　　　　　ميزة - خاصية

Abstraction: Hiding the internal implementation of the (method, feature) and only showing the functionality to the users.



```
void method1 ( ){

        ..
        ..
        ..
        ..

}
```

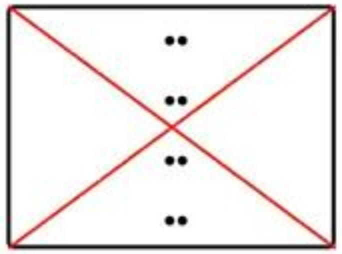
Abstraction

What is the Abstraction

ميزة - خاصية   إخفاء التنفيذ الداخلي

Abstraction: Hiding the internal implementation of the (method, feature) and only showing the functionality to the users.

void method1 ( ){ $\longrightarrow$ Service



}

Abstraction

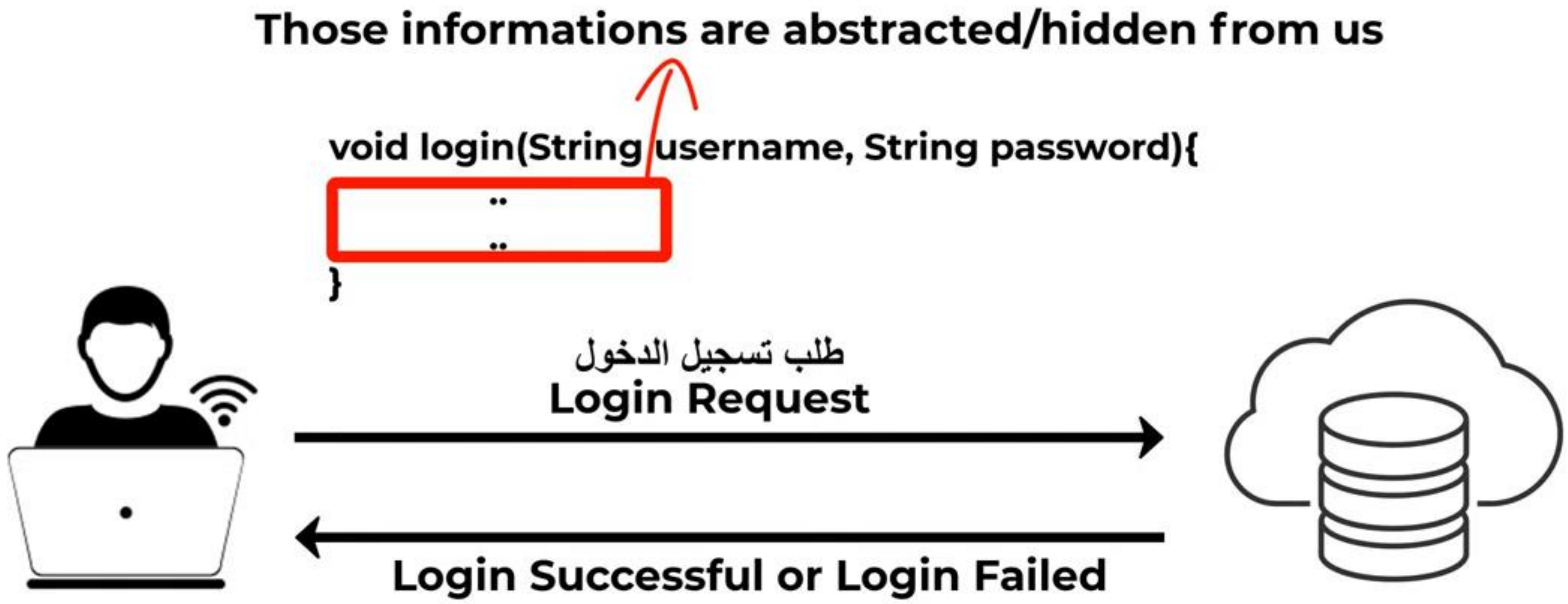
What is the Abstraction

إخفاء التنفيذ الداخلي

ميزة - خاصية

Abstraction: **Hiding the internal implementation** of the (**method**, **feature**) and **only showing the functionality** to the users.

void method1 ( ); $\longrightarrow$ Service


Those informations are abstracted/hidden from us
طلب تسجيل الدخول
Login Request
Login Successful or Login Failed

```
void login(String username, String password){
                ..
                ..
}
```

Real-Life Example

Abstraction

Types of Abstraction:

- **Data Abstraction**
- **Control Abstraction**

Ways to achieve Abstraction:

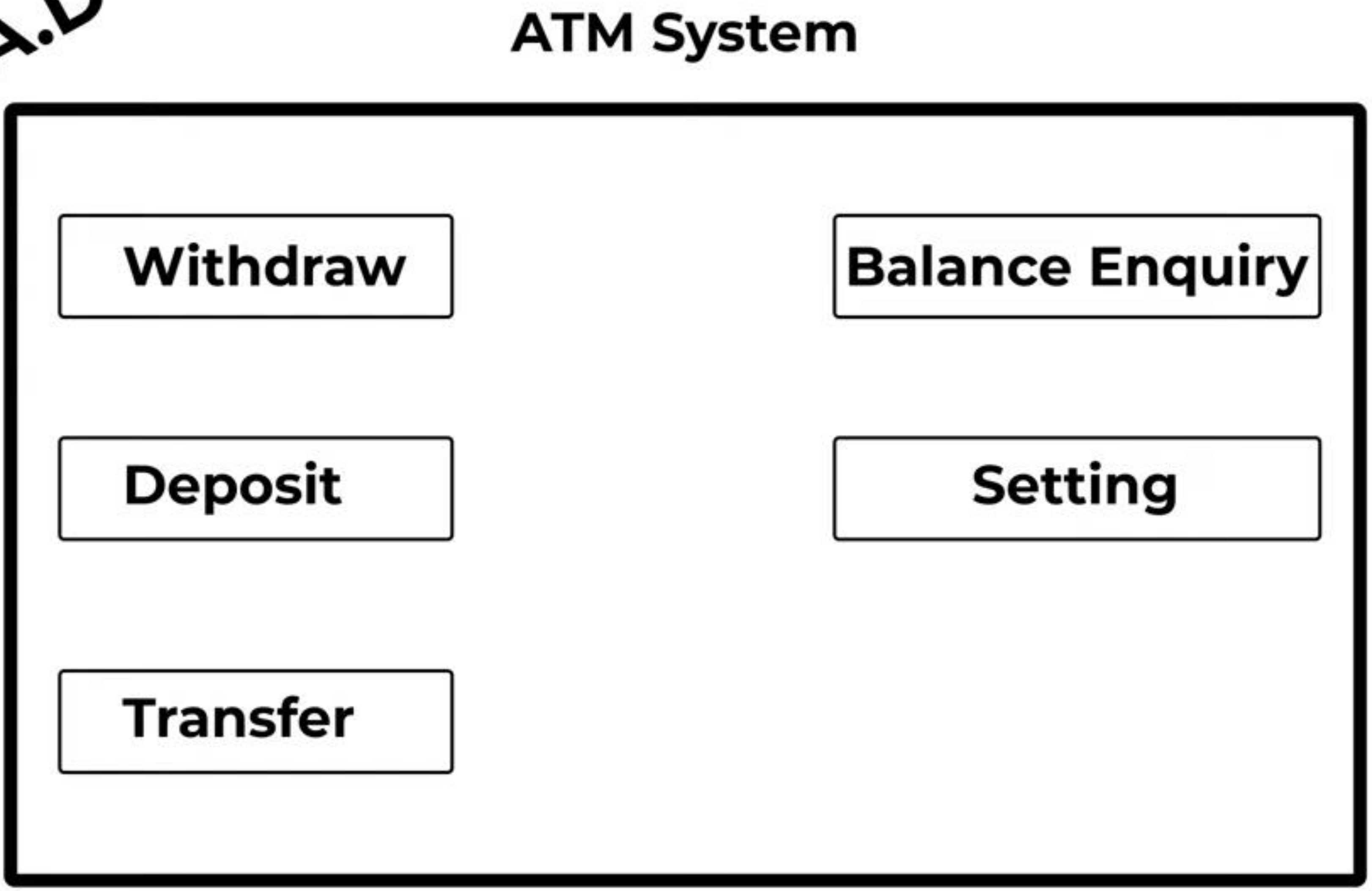
- **Abstract Class**
- **Interface**

Abstract Class

What Is Abstract Class?
Abstract classes allow you to create **blueprints** for **concrete classes**.
But the inheriting class should implement the abstract method.

Abstract Class

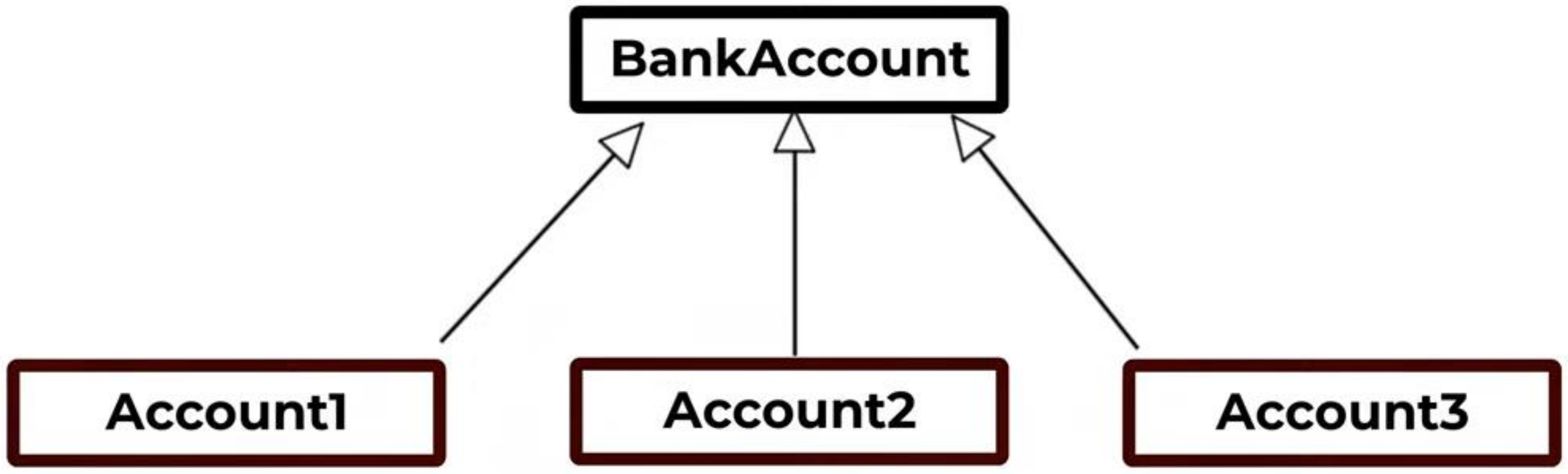
OO.A.D



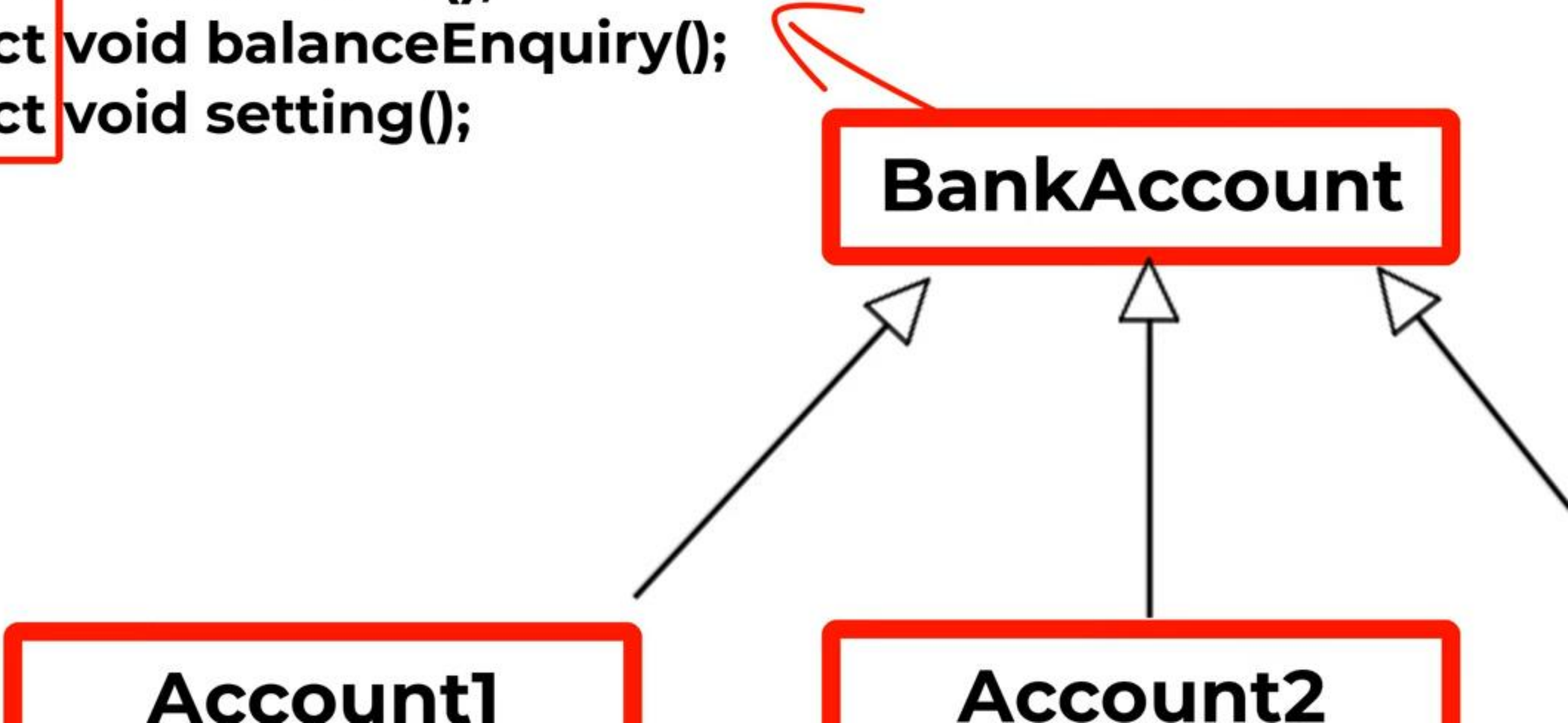
```
public abstract class BankAccount{

    abstract void withdraw();
    abstract void deposit();
    abstract void transfer();
    abstract void balanceEnquiry();
    abstract void setting();

}
```

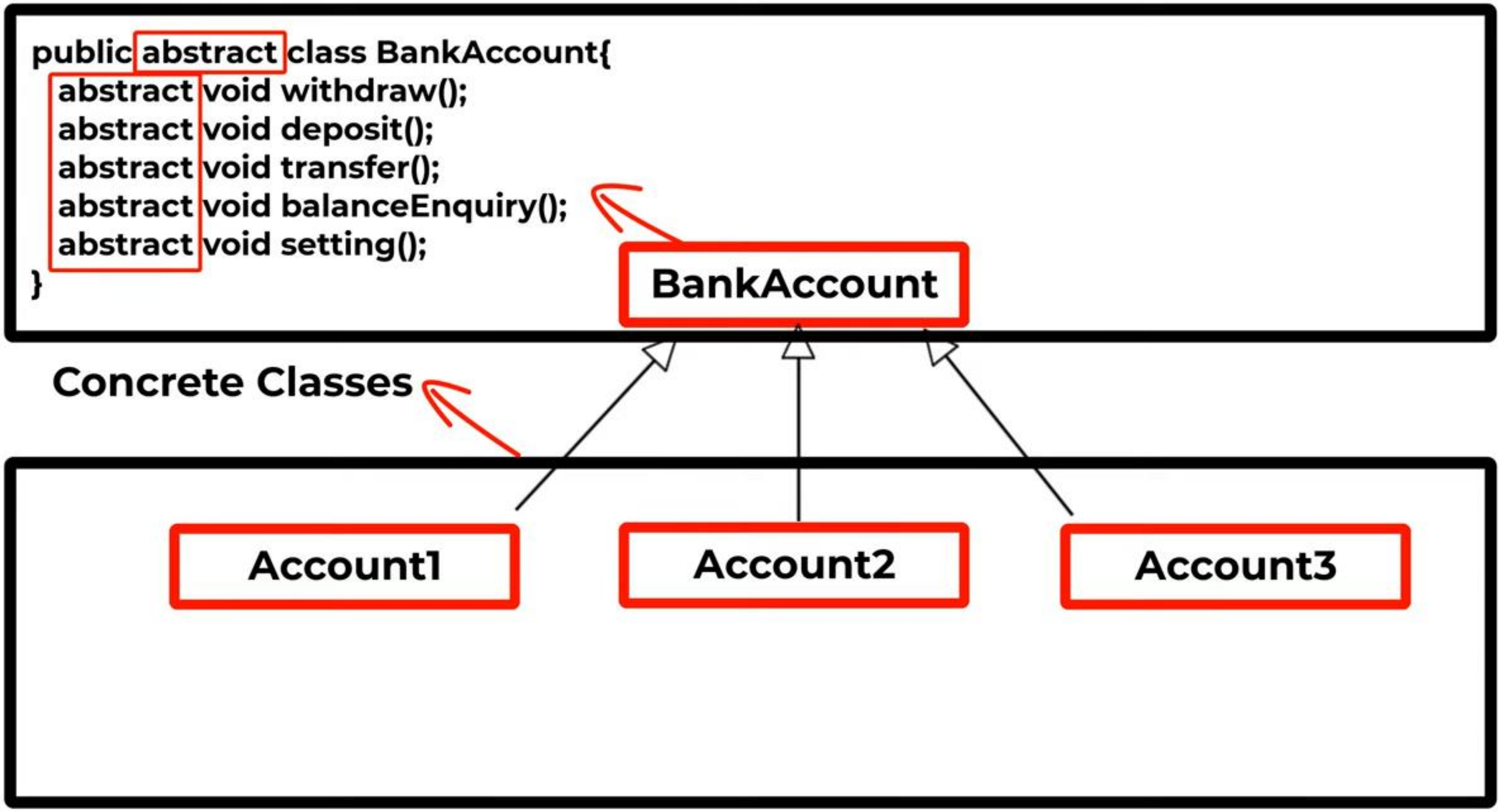
any concrete bank account class that extends it must provide implementations for all these methods.

Abstract Class Is Always Parent



```
public abstract class BankAccount{
  abstract void withdraw();
  abstract void deposit();
  abstract void transfer();
  abstract void balanceEnquiry();
  abstract void setting();
}
```




```java
public abstract class BankAccount{
  abstract void withdraw();
  abstract void deposit();
  abstract void transfer();
  abstract void balanceEnquiry();
  abstract void setting();
}
```

Account1

```
public class Account1{
  @Override
  void withdraw(){
      ..
  }
  @Override
  void deposit(){
      ..
  }
  @Override
  void transfer(){
      ..
  }
  @Override
  void balanceEnquiry(){
      ..
```

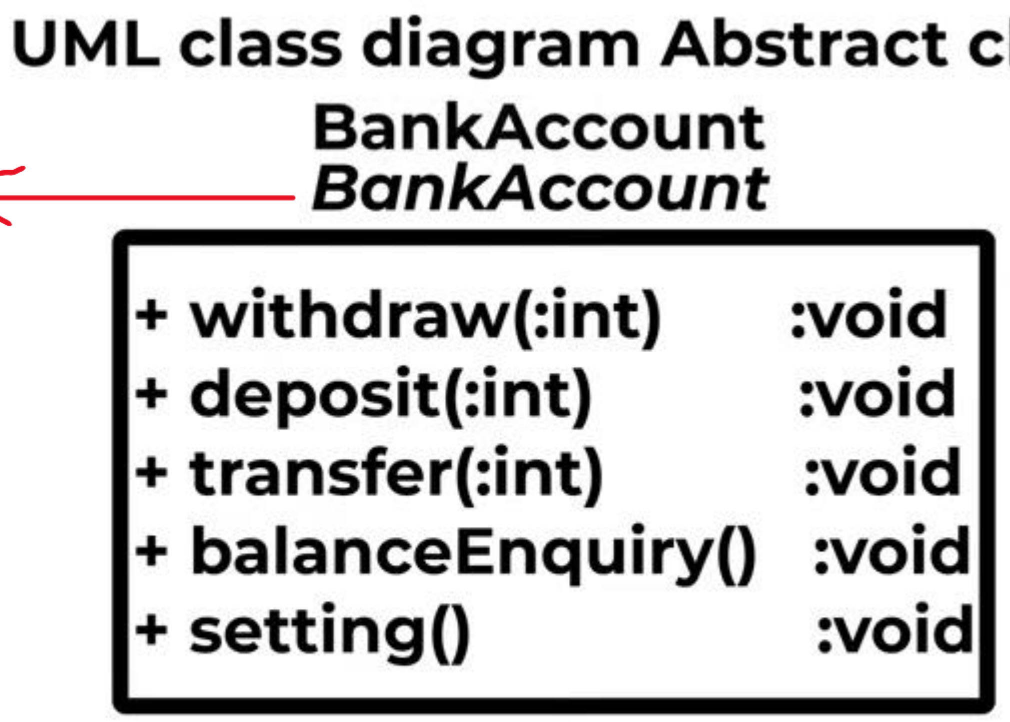
```
public class Account1{
  void withdraw(){
      ..
  }
  void deposit(){
      ..
  }
  void transfer(){
      ..
  }
  void balanceEnquiry(){
      ..
  }
  void setting(){
      ..
  }
}
```

A **concrete class** is a class that has an implementation for all of its methods.

It is a non-access modifier

Abstract class in Java

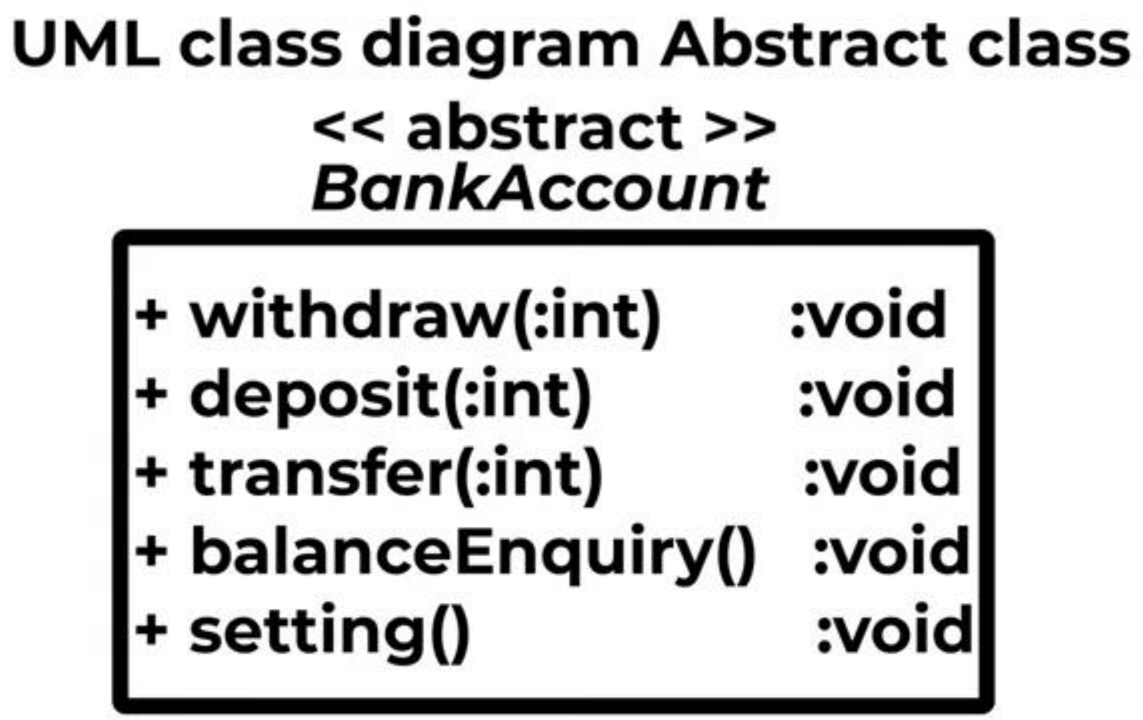
A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).



in *italics*

you can describe an **abstract class** using **either** of these two standard ways

<< abstract >>



you can describe an **abstract**
class using **either** of these two standard ways

Rules for Abstract Class:

- An abstract class must be declared with an **abstract keyword**.
- It can have **abstract** and **non-abstract** methods.
- It **cannot be instantiated**.
- It can have **final methods** which will force the subclass not to change the body of the method.
- It can have **constructors** and **static methods** also.